Acronym-Meaning Extraction from Corpora Using Multitape Weighted Finite-State Machines

— Research Report 2006 / 019 —

André Kempe

Xerox Research Centre Europe – Grenoble Laboratory
6 chemin de Maupertuis – 38240 Meylan – France
andre.kempe@xrce.xerox.com – http://www.xrce.xerox.com

July 26, 2006

Abstract

The automatic extraction of acronyms and their meaning from corpora is an important sub-task of text mining. It can be seen as a special case of string alignment, where a text chunk is aligned with an acronym. Alternative alignments have different cost, and ideally the least costly one should give the correct meaning of the acronym.

We show how this approach can be implemented by means of a 3-tape weighted finite-state machine (3-WFSM) which reads a text chunk on tape 1 and an acronym on tape 2, and generates all alternative alignments on tape 3. The 3-WFSM can be automatically generated from a simple regular expression. No additional algorithms are required at any stage. Our 3-WFSM has a size of 27 states and 64 transitions, and finds the best analysis of an acronym in a few milliseconds.

1 Introduction

The automatic extraction of acronyms and their meaning from corpora is an important sub-task of text mining. We will refer to it by the term *acronym-meaning extraction*.

Much work has been done on it. To mention just some: Yeates, Bainbridge, and Witten (2000) matched an acronym against a text chunk, thus producing different candidate definitions for the acronym. They alternatively tried heuristic approaches, naive Bayes learning, and compression (i.e., shortest description length) to select the best candidate. Pustejovsky et al. (2001) added shallow parsing, and matched an acronym against a parsed (i.e., structured) text chunk. Schwartz and Hearst (2003) used a heuristic algorithm to deterministically find an acronym's meaning.

The task can bes seen as a special case of string alignment between a text chunk and an acronym. For example, the text chunk "they have many hidden Markov models" can be aligned with the acronym "HMMs" in different ways, such as "they have many **h**idden **M**arkov **m**odel**s**" or "t**h**ey have **m**any hidden **M**arkov model**s**". Alternative alignments have different cost, and ideally the least costly one should give the correct meaning of the acronym. String alignment admits in general four different edit operations: insertion, deletion, substitution, and preservation of a letter (Wagner and Fischer, 1974; Pirkola et al., 2003). In the case of acronym-meaning extraction, only deletion and preservation can occur. String alignment has a worst-case time complexity of $\mathcal{O}(|s_1||s_2|)$, with $|s_1|$ and $|s_2|$ being the lengths of the two aligned strings, respectively. This also holds for the present special case.

The purpose of this report is to show how the alignment-based approach to acronym-meaning extraction can be implemented by means of a 3-tape weighted finite-state machine (3-WFSM). The 3-WFSM will read a text chunk on tape 1 and an acronym on tape 2, and generate all possible alignments on tape 3, inserting dots to mark which letters are used in the acronym. For the above example this would give "they have many .hidden .Markov .model.s", among others.

The 3-WFSM can be automatically generated from a clear and relatively simple regular expression that defines the task in as much detail as we wish. Both the generation and the application of the 3-WFSM are done by (more or less) standard operations (Kempe, Champarnaud, and Eisner, 2004), that are available in finite-state tools such as WFSC (Kempe et al., 2003). No additional algorithms are required at any stage, which reduces the development time to a few hours.

2 Preprocessing the Corpus

Using basic UNIX utilities, we first extract from a corpus all sentences that contain acronyms in brackets, such as

Between a hidden Markov model (HMM) and a weighted finite-state
  automaton (WFSA) there is a correspondence.

Then, we split these sentences into pairs consisting of an acronym and the text chunk that precedes it (starting from the sentence beginning or form the preceding acronym, respectively). For the above example, this is

Between a hidden Markov model	HMM
and a weighted finite-state automaton	WFSA

Next, we normalize these pairs: capital letters are transformed into small ones, and separators into underscores:

between_a_hidden_markov_model	hmm
and_a_weighted_finite_state_automaton	wfsa

3 Constructing an Acronym-Meaning Extractor

We start by compiling a 4-WFSM over the real tropical semiring $\langle \mathbb{R}_{\geq 0} \cup \{\infty\}, \min, +, \infty, 0 \rangle$, from the expression

$$A_1^{(4)} = \Big(\Big\langle \langle \varepsilon, \varepsilon, \ . \ , \varepsilon \rangle, 0 \Big\rangle \Big\langle \langle \texttt{[\^{}\_]}, \texttt{[\^{}\_]}, \texttt{[\^{}\_]}, \texttt{a} \rangle_{\{1=2=3\}}, 0 \Big\rangle \ \cup \ \Big\langle \langle \texttt{[\^{}\_]}, \varepsilon, \texttt{[\^{}\_]}, \texttt{i} \rangle_{\{1=3\}}, 0 \Big\rangle \ \cup \ \Big\langle \langle \_, \varepsilon, \_, \_ \rangle, 0 \Big\rangle \Big)^* \qquad (1)$$

where [^_] is a *symbol class* accepting any symbol except underscore, ε represents the empty string, $\{1\!=\!2\!=\!3\}$ a constraint requiring the different [^_] on tapes 1 to 3 to be instantiated by the same symbol (Nicart et al., 2006),[1] and the 0's are weights. We use a superscript $^{(n)}$ to indicate the arity of a n-WFSM (Kempe, Champarnaud, and Eisner, 2004).

If we apply $A_1^{(4)}$ with tapes 1 and 2 to a normalized text chunk and a corresponding acronym, respectively, and generate (neutrally-weighted) alternative analyses from tapes 3 and 4, we obtain, for example

```
1,2> they_have_many_hidden_markov_models      hmms

3,4> t.hey_have_.many_hidden_.markov_model.s    iaii_iiii_aiii_iiiiii_aiiiii_iiiiia     0
3,4> t.hey_have_.many_hidden_markov_.model.s    iaii_iiii_aiii_iiiiii_iiiiii_aiiiia     0
3,4> t.hey_have_many_hidden_.markov_.model.s    iaii_iiii_iiii_iiiiii_aiiiii_aiiiia     0
3,4> they_.have_.many_hidden_.markov_model.s    iiii_aiii_aiii_iiiiii_aiiiii_iiiiia     0
3,4> they_.have_.many_hidden_markov_.model.s    iiii_aiii_aiii_iiiiii_iiiiii_aiiiia     0
3,4> they_.have_many_hidden_.markov_.model.s    iiii_aiii_iiii_iiiiii_aiiiii_aiiiia     0
3,4> they_have_many_.hidden_.markov_.model.s    iiii_iiii_iiii_aiiiii_aiiiii_aiiiia     0
```

On tape 3, letters of the text chunk which are used in the acronym, are preceded by a dot. Tape 4 shows the performed operations: `a` meaning "acronym letter", `i` meaning "ignored letter". All analyses have neutral weight 0.

By means of XFST (Karttunen, Gaál, and Kempe, 1998; Beesley and Karttunen, 2003) we generate from the following regular expression a 2-FSM (i.e., a non-weighted transducer), ${A'_2}^{(2)}$, that defines the operations more precisely:

```
regex
    [%_|a|u|i|g|G|1|2|3|4|5|6|7|8]*
.o. [ i -> u ||  a [ i | %_ ]* %_         _  i* [ .#. | %_ ] ]
.o. [ i -> g ||                           _  i* a  ]
.o. [ g -> G ||  [ .#. | %_ ]             _  ]
.o. [ a -> 1 ||  [ .#. | %_ ]             _  ]
.o. [ a -> 2 ||  [ .#. | %_ ] \%_         _  ]
.o. [ a -> 3 ||  [ .#. | %_ ] \%_^2       _  ]                    (2)
.o. [ a -> 4 ||  [ .#. | %_ ] \%_^3       _  ]
.o. [ a -> 5 ||  [ .#. | %_ ] \%_^4       _  ]
.o. [ a -> 6 ||  [ .#. | %_ ] \%_^5       _  ]
.o. [ a -> 7 ||  [ .#. | %_ ] \%_^6       _  ]
.o. [ a -> 8 ||  [ .#. | %_ ] \%_^7 \%_*  _  ]
.o. [%_|a|u|i|g|G|1|2|3|4|5|6|7|8]*
```

In this expression, all word-initial `i` are replaced by `u` ("unused word") if no letter of this word is used in the acronym, but if letters of preceding words are used. Then, all `i` of a used word are replaced by `g` ("gap letter") if they are followed by an `a` ("acronym letter") in the same word. Next, word-initial `g` are replaced by `G` ("word-initial gap letter"). Finally, all `a` are replaced by a number `1` to `8`, expressing their position in the word. Positions higher than 8 are marked as `8`.

Furthermore, we generate with XFST another 2-FSM, ${A'_3}^{(2)}$, that deletes all letters of leading unused words, and the adjacent underscores:

```
regex
    [%.|%_|a|b|c|d|e|f|g|h|i|j|k|l|m|n|o|p|q|r|s|t|u|v|w|x|y|z]*
.o. [ \%. -> 0 || .#. \%.* _ \%.* %_ ]                            (3)
.o. [  %_ -> 0 || .#. _ ]
.o. [%.|%_|a|b|c|d|e|f|g|h|i|j|k|l|m|n|o|p|q|r|s|t|u|v|w|x|y|z]*
```

[1] Roughly following (Kempe, Champarnaud, and Eisner, 2004), we employ here a simpler notation for constraints than in (Nicart et al., 2006).

These two non-weighted 2-FSMs are transformed into 2-WFSMs, $A_2^{(2)}$ and $A_3^{(2)}$, with neutral weight, and are joined (Kempe, Champarnaud, and Eisner, 2004) with the previously compiled 4-WFSM $A_1^{(4)}$:

$$A_4^{(6)} = \left(A_1^{(4)} \bowtie_{\{3=1\}} A_3^{(2)} \right) \bowtie_{\{4=1\}} A_2^{(2)} \tag{4}$$

In the resulting $A_4^{(6)}$, we have a modified form of tape 3 on tape 5 (describing analyses), and a modified form of tape 4 on tape 6 (describing operations). If we apply tape 1 to a normalized text chunk and tape 2 to a corresponding acronym, we obtain from tapes 5 and 6 for example

```
1,2> they_have_many_hidden_markov_models     hmms

5,6> t.hey_have_many_hidden_.markov_.model.s   G2ii_uiii_uiii_uiiiii_1iiiii_1gggg6     0
5,6> t.hey_have_.many_hidden_markov_.model.s   G2ii_uiii_1iii_uiiiii_uiiiii_1gggg6     0
5,6> t.hey_have_.many_hidden_.markov_model.s   G2ii_uiii_1iii_uiiiii_1iiiii_Ggggg6     0
5,6>      .have_many_hidden_.markov_.model.s   iiii_1iii_uiii_uiiiii_1iiiii_1gggg6     0
5,6>      .have_.many_hidden_markov_.model.s   iiii_1iii_1iii_uiiiii_uiiiii_1gggg6     0
5,6>      .have_.many_hidden_.markov_model.s   iiii_1iii_1iii_uiiiii_1iiiii_Ggggg6     0
5,6>                .hidden_.markov_.model.s   iiii_iiii_iiii_1iiiii_1iiiii_1gggg6     0
```

Finally, we assign costs (i.e., weights) to the different operations by means of a 1-WFSM generated from the expression

$$\begin{aligned} A_5^{(1)} \quad = \quad & (\langle \_, 0\rangle \cup \langle \mathtt{i}, 0\rangle \cup \langle \mathtt{u}, 2\rangle \cup \langle \mathtt{g}, 1\rangle \cup \langle \mathtt{G}, 3\rangle \cup \\ & \langle \mathtt{1}, 0\rangle \cup \langle \mathtt{2}, 1\rangle \cup \langle \mathtt{3}, 1.5\rangle \cup \langle \mathtt{4}, 2\rangle \cup \langle \mathtt{5}, 2.5\rangle \cup \langle \mathtt{6}, 3\rangle \cup \langle \mathtt{7}, 3.5\rangle \cup \langle \mathtt{8}, 4\rangle) \end{aligned} \tag{5}$$

Here we chose the costs by intuition. In an improved approach they could be estimated from data.

To obtain our acronym-meaning extractor, we join $A_5^{(1)}$ with the previously compiled $A_4^{(6)}$:

$$\mathsf{Acro}^{(6)} = A_4^{(6)} \bowtie_{\{6=1\}} A_5^{(1)} \tag{6}$$

If we apply $\mathsf{Acro}^{(6)}$ with tapes 1 and 2 to a text chunk and a corresponding acronym, respectively, we obtain from tapes 5 and 6 for example

```
1,2> they_have_many_hidden_markov_models     hmms

5,6> t.hey_have_many_hidden_.markov_.model.s   G2ii_uiii_uiii_uiiiii_1iiiii_1gggg6     17
5,6> t.hey_have_.many_hidden_markov_.model.s   G2ii_uiii_1iii_uiiiii_uiiiii_1gggg6     17
5,6> t.hey_have_.many_hidden_.markov_model.s   G2ii_uiii_1iii_uiiiii_1iiiii_Ggggg6     18
5,6>      .have_many_hidden_.markov_.model.s   iiii_1iii_uiii_uiiiii_1iiiii_1gggg6     11
5,6>      .have_.many_hidden_markov_.model.s   iiii_1iii_1iii_uiiiii_uiiiii_1gggg6     11
5,6>      .have_.many_hidden_.markov_model.s   iiii_1iii_1iii_uiiiii_1iiiii_Ggggg6     12
5,6>                .hidden_.markov_.model.s   iiii_iiii_iiii_1iiiii_1iiiii_1gggg6      7
```

We select the analysis with the lowest cost by means of a classical single-source best-path algorithm such as Dijkstra's algorithm (Dijkstra, 1959) or Bellman-Ford's (Bellman, 1958; Ford and Fulkerson, 1956). If our weights have been optimally chosen, we should now obtain the correct analysis.

In practice, input is read on tapes 1 and 2 and output generated from tapes 2 and 5, as in the following examples. All other tapes can therefore be removed, leaving us with a 3-WFSM $\mathsf{Acro}^{(3)}$.

```
1,2> they_have_many_hidden_markov_models    hmms
1,2> between_hidden_markov_models           hmms
1,2> and_weighted_finite_state_automata     wfsa
1,2> and_weighted_finite_state_automata     wfa

2,5> hmms    .hidden_.markov_.model.s
2,5> hmms    .hidden_.markov_.model.s
2,5> wfsa    .weighted_.finite_.state_.automata
2,5> wfa     .weighted_.finite_state_.automata
```

4 Some Results

We tested the acronym-meaning extractor on many examples. Finding the best analysis for one acronym took only a few milliseconds (ms) : For example, 3.7 ms for “they have many hidden markov models”-“hmms”, 9.6 ms for “they have many hidden markov models they have many hidden markov models”-“hmms”, and 12.0 ms for “they have many hidden markov models they have many hidden markov models”-“hmmshmms”.

The extractor had a size of 27 states and 64 transitions.

References

Beesley, Kenneth R. and Lauri Karttunen. 2003. *Finite State Morphology*. CSLI Publications, Palo Alto, CA.

Bellman, Richard. 1958. On a routing problem. *Quarterly of Applied Mathematics*, 16:87–90.

Dijsktra, Edsger W. 1959. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271.

Ford, Lester R. and Delbert R. Fulkerson. 1956. Maximal flow through a network. *Canadian Journal of Mathematics*, 8(3):99–404.

Karttunen, Lauri, Tamás Gaál, and André Kempe. 1998. *Xerox finite state complier*. Online demo and documentation. Xerox Research Centre Europe, Grenoble, France. http://www.xrce.xerox.com/competencies/content-analysis/fsCompiler/.

Kempe, André, Christof Baeijs, Tamás Gaál, Franck Guingne, and Florent Nicart. 2003. WFSC – A new weighted finite state compiler. In O. H. Ibarra and Z. Dang, editors, *Proc. 8th Int. Conf. CIAA*, volume 2759 of *Lecture Notes in Computer Science*, pages 108–119, Santa Barbara, CA, USA. Springer Verlag, Berlin, Germany.

Kempe, André, Jean-Marc Champarnaud, and Jason Eisner. 2004. A note on join and auto-intersection of *n*-ary rational relations. In B. Watson and L. Cleophas, editors, *Proc. Eindhoven FASTAR Days*, number 04–40 in TU/e CS TR, pages 64–78, Eindhoven, Netherlands.

Nicart, Florent, Jean-Marc Champarnaud, Tibor Csáki, Tamás Gaál, and André Kempe. 2006. Multi-tape automata with symbol classes. In O.H. Ibarra and H.-C. Yen, editors, *Proc. 11th Int. Conf. on Implementation and Application of Automata (CIAA’06)*, volume 4094 of *Lecture Notes in Computer Science*, pages 126–136, Taipei, Taiwan. Springer Verlag.

Pirkola, Ari, Jarmo Toivonen, Heikki Keskustalo, Kari Visala, and Kalervo Järvelin. 2003. Fuzzy translation of cross-lingual spelling variants. In *Proceedings of the 26th Annual International ACM SIGIR*, pages 345–352, Toronto, Canada.

Pustejovsky, James, José Casta no, Brent Cochran, Maciej Kotecki, Michael Morrell, and Anna Rumshisky. 2001. Linguistic knowledge extraction from medline: Automatic construction of an acronym database. In *Proc. 10th World Congress on Health and Medical Informatics (Medinfo 2001)*.

Schwartz, Ariel and Marti Hearst. 2003. A simple algorithm for identifying abbreviation definitions in biomedical texts. In *Proc. Pacific Symposium on Biocomputing (PSB-2003)*.

Wagner, Robert A. and Michael J. Fischer. 1974. The string-to-string correction problem. *Journal of the Association for Computing Machinery*, 21(1):168–173.

Yeates, Stuart, David Bainbridge, and Ian H. Witten. 2000. Using compression to identify acronyms in text. In *Proc. Data Compression Conference (DCC-2000)*, Snowbird, Utah, USA. (Also published in a longer form as Working Paper 00/01, Department of Computer Science, University of Waikato, January 2000).